MEKONG UNIVERSITY SCIENTIFIC JOURNAL

https://tapchikhdhcl.edu.vn/mkusj/ P-ISSN 3093-3218
https://vjol.info.vn/dhcl-e/index E-ISSN 3093-3358

EXPLORING THE POTENTIAL OF CONVOLUTIONAL NEURAL NETWORKS FOR PHISHING URL DETECTION IN CYBERSECURITY

Tran Thị Thuy*
Mekong University
*Email: tranthithuy@mku.edu.vn

DOI: https://doi.org/10.65934/mkusj.2026.02.1142
Received: 14/03/2026 *Reviewed: 22/04/2026* *Accepted: 28/06/2026*

ABSTRACT

Phishing attacks continue to pose one of the most significant cybersecurity threats by exploiting deceptive websites and malicious URLs to obtain sensitive personal and organizational information. As phishing techniques become increasingly sophisticated, accurate and automated detection methods are essential for enhancing cybersecurity protection. This study investigates the potential of Convolutional Neural Networks (CNNs) for phishing URL detection by leveraging their capability to automatically extract discriminative features from sequential textual data.The proposed approach incorporates data preprocessing and hyperparameter optimization to improve the performance of the CNN-based classification model. The model was trained and evaluated using a dataset containing both phishing and legitimate URLs. Experimental results demonstrate that the proposed model achieved a classification accuracy of 85.42%, indicating that CNNs provide an effective approach for distinguishing phishing URLs from legitimate ones. The findings suggest that deep learning techniques, particularly CNNs, can enhance automated phishing detection while reducing the need for manual feature engineering. This study contributes to the development of intelligent cybersecurity solutions and provides a foundation for future research on improving phishing detection through advanced deep learning architectures and optimization techniques.

Keywords: *phishing URL detection; convolutional neural networks; cybersecurity; deep learning; hyperparameter optimization.*

Introduction

Phishing attacks, which exploit deception to steal sensitive information, are becoming increasingly sophisticated and difficult to detect, especially when they use fake websites. This highlights the need for effective detection methods to protect personal and organizational data. Phishing remains a major threat in cybersecurity, requiring robust and accurate detection mechanisms.

Convolutional Neural Networks have proven effective in text analysis and spam detection tasks. CNNs are particularly powerful in extracting features from sequential data such as URLs, enabling the detection of phishing attacks. This study applies CNNs to automatically detect phishing URLs from raw data.

The paper is organized as follows: Section 2 reviews related work, Section 3 describes the research methodology, Section 4 presents the experiments, Section 5 compares the results using standard metrics, and Section 6 provides conclusions and suggests directions for future research.

1. Related Works

In many studies, traditional machine learning techniques have been used for phishing detection, such as Support Vector Machines (SVMs) (Gupta et al.), Decision Trees, and Random Forests (Khan et al.), to classify URLs based on features such as domain names, URL length, and character patterns. For example, SVM combined with URL-based features was used by Gupta et al., achieving an accuracy of approximately 70%. Random Forests were applied by Khan et al., and accuracies ranging from 65% to 72% were reported. However, these methods are often dependent on handcrafted features, which limits the ability to identify complex patterns.

Deep learning approaches, particularly neural network architectures such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, have been extensively applied to phishing detection. Sequential dependencies

in URLs were captured using LSTM by Zhang et al., achieving an accuracy of 74%. BiLSTM was employed by Lee et al., and improved performance compared to traditional approaches was achieved.

Recently, Convolutional Neural Networks (CNNs) have been increasingly investigated due to their ability to automatically extract features from raw data, enabling more effective phishing detection than traditional methods. The effectiveness of CNNs in phishing URL classification was demonstrated by Huang et al., with high accuracy being achieved and significant performance improvements over previous approaches being reported.

Hyperparameter tuning has been recognized as a critical factor in optimizing deep learning models. Traditional methods such as Grid Search and Random Search, while effective, are considered time-consuming and resource-intensive (Zhang et al.). Recently, Keras Tuner has been introduced to allow faster and more efficient hyperparameter optimization (Chollet et al.). The importance of hyperparameter tuning in achieving optimal results and improving model accuracy was emphasized by Zhang et al.

2. Research Methodology

Data Collection and Preprocessing

The dataset used in this study consists of URLs labeled as phishing or legitimate and was obtained from a public dataset. The preprocessing procedure was carried out through the following main steps:

(1) Data cleaning was performed to remove duplicate and irrelevant entries, ensuring that the dataset accurately represents phishing and legitimate URLs.

(2) Label encoding was applied using the LabelEncoder from scikit-learn to convert categorical labels into numerical values.

(3) Text tokenization was conducted using the Keras Tokenizer to transform URLs into sequences of integers.

(4) Sequence padding was applied using the Keras pad_sequences function to standardize the length of the sequences, ensuring a uniform input size for the deep learning model.

CNN Model Architecture

The CNN model was initialized with an Embedding Layer, in which integer sequences were transformed into dense vectors, allowing relationships and semantic information among URL components to be captured. Subsequently, a Convolutional Layer was employed, where filters were used to extract salient features from URLs, enabling the model to learn local patterns characteristic of phishing or legitimate URLs. A Max Pooling Layer was applied to reduce computational complexity and prevent overfitting, while retaining important features. A Flatten Layer was then used to convert feature maps into one-dimensional vectors in preparation for the classification layers. Dense Layers were utilized to aggregate the extracted features and perform the final classification. Finally, an Output Layer with a sigmoid activation function was used to return the probability of a URL being phishing or legitimate (Figure 1).

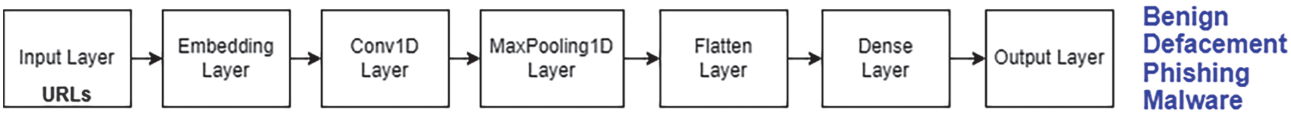

Figure 1. CNN Model Architecture

Hyperparameter Tuning

Multifidelity optimization techniques have been recognized as important for addressing challenges related to space, time, and computational resources in machine learning models, particularly under budget constraints. One effective method in this domain is Hyperband (Li et al.), which has been widely adopted due to its efficiency in hyperparameter tuning. Hyperband combines bandit-based algorithms with random search to improve model performance over multiple iterations. This method operates by eliminating poorly performing hyperparameter configurations and allocating more resources to promising ones, thereby optimizing the training process. Budget constraints and hyperparameter configurations were defined in advance based on the size of the input data and the minimum number of training samples, allowing the number of configurations and the budget size for each trial to be adjusted accordingly

Hyperband Algorithm

Input: Maximum budget b_max, Minimum budget b_min

1. Computation of the maximum number of iterations

$$S_{max} = \log\log\frac{b_{max}}{b_{min}}$$

2. $for\ s\{b_{max}, b_{min} - 1, \ldots, 0$ the following steps are performed:

2.1. Determination of the number of configurations

$$n = DetermineBudget(s)$$

2.2. Sampling of hyperparameter configurations

$$\gamma = SampleConfiguration(n)$$

2.3. Application of the Successive Halving algorithm

$$SuccessiveHalving(\gamma)$$

3. Return of the best configuration:
After all iterations are completed, the configuration with the best performance is selected.

Hyperband employs the Successive Halving technique to gradually eliminate poorly performing configurations and allocate larger budgets to more promising ones (Li et al.). This process enables fast and efficient hyperparameter optimization while reducing computational cost. It is particularly beneficial for optimizing complex models such as CNNs, improving both efficiency and accuracy in phishing URL detection.

Successive Halving Algorithm

Input: Number of configurations n, Total budget B

1. Initialization of budget allocation:
   • The initial budget for each configuration is set as:

$$b = \frac{B}{n}$$

2. Configuration refinement through iterative steps:
   2.1. For each iteration iii:
   • Performance evaluation is conducted by allocating a budget bib_ibi to each configuration and assessing its performance.
   2.2. Elimination of underperforming configurations:
   • Fifty percent of the lowest-performing configurations are discarded based on their evaluation results.
   2.3. Increase of budget for remaining configurations:

$$b_{i+1} = 2 \times b_i$$

3. Continuation until a single configuration remains:
   This iterative process continues until the optimal configuration is identified.

Model Training and Evaluation

The CNN model was trained using optimization techniques to achieve high performance and to reduce overfitting. An early stopping technique was applied to terminate training when performance on the validation set was no longer improved, thereby preventing overfitting and enhancing generalization capability. Learning rate reduction was employed to adaptively adjust the learning rate when further improvement was not observed, allowing the model to stabilize and optimize its performance. Finally, evaluation metrics such as accuracy and the loss function were used to measure model effectiveness, ensuring an objective and reliable assessment on the test set.

3. Experiments

Experimental Dataset

- The Malicious URLs Dataset was used in the experiments (Mishra et al.), consisting of a total of 651,191 URLs, which were categorized into four groups based on their threat levels:
- **Benign URLs:** 428,103 safe URLs with no associated risk.
- **Defacement URLs:** 96,457 URLs related to malicious website defacement.
- **Phishing URLs:** 94,111 URLs used in phishing attacks, in which legitimate websites are impersonated to deceive users.

Malware URLs: 32,520 URLs involved in the distribution of malicious software.

The dataset contains two columns: *"url"*, which includes the web addresses to be analyzed, and *"type"*, which specifies the classification of each URL. These labels are essential for supervised learning, as they enable the distinction between legitimate and malicious URLs. This dataset is considered a valuable resource for training and evaluating malicious URL detection models, as the diversity of malicious URLs allows multiple types of threats to be learned. However, the dataset presents a class imbalance challenge, in which benign URLs dominate, requiring special handling techniques to ensure effective model training and evaluation.

Hyperparameter-Tuned Convolutional Neural Network (CNN) for URL Classification

The CNN model used in this study was composed of a sequence of essential layers. First, an input layer was used to receive text data of a fixed length and feed it into the model. Next, an embedding layer was employed to transform text sequences into dense vectors, with configuration parameters such as vocabulary size (*input_dim*) and vector dimension (*output_dim*). Subsequently, a one-dimensional convolutional layer was applied, in which filters were used to learn spatial features from textual data, with adjustable filter sizes and numbers of filters. A MaxPooling1D layer was then utilized to reduce data dimensionality, retain important features, and decrease spatial complexity.

A flatten layer was used to convert the three-dimensional tensor output into a one-dimensional vector, enabling connection to fully connected (dense) layers. The first dense layer was employed to perform high-level inference and feature extraction, with the ReLU activation function being applied to learn complex patterns. To prevent overfitting, a dropout layer was optionally added, in which a proportion of units was randomly omitted during training. In addition, a second dense layer could be included to learn more complex features. Finally, an output layer with a single unit and a sigmoid activation function was used to produce the probability of a URL being classified as phishing.

This model enabled accurate URL classification while providing robustness against overfitting and optimized learning performance.

4. Results

An accuracy of 85.42% was achieved by the proposed CNN model, with optimal hyperparameters being identified through the Hyperband algorithm, as presented in Table 1 and Figure 2. These results demonstrate that effective discrimination between phishing URLs and legitimate URLs was achieved. The learning curve (Figure 3) indicates that model performance improved across epochs, with test accuracy stabilizing as training was completed.

To further evaluate effectiveness, the CNN model was compared with existing phishing detection approaches:

Gupta et al. applied an SVM-based approach and achieved an accuracy of approximately 70%. Although SVM was shown to be effective for phishing detection, careful handcrafted feature engineering was required, and difficulties may arise in identifying complex patterns without extensive preprocessing.

Khan et al. employed Recurrent Neural Networks (RNNs) and reported accuracies ranging from 72% to 78%. While RNNs are capable of handling sequential data and capturing temporal dependencies, training difficulties may be encountered, and gradient vanishing issues can arise when long sequences are processed.

Table 1. Hyperparameters of the CNN Model Optimized Using the Hyperband Algorithm

Parameters	embedding_dim	filters	kernel_size	pool_size	dense_units	learning_rate	val_accuracy	Total elapsed time	Best val_accuracy So Far
Running Trial #1	64	32	3	3	64	0.0019318	-	-	-
Running Trial #2	32	64	5	3	32	0.001687	**0.725000024**	00h 05m 21s	**0.725000024**
Running Trial #3	32	48	5	3	96	0.00049938	0.689999998	00h 05m 24s	0.725000024
Running Trial #4	32	64	5	2	64	0.00417118	0.689999998	00h 05m 27s	0.725000024
Running Trial #5	32	64	3	2	96	0.0047288	0.694999993	00h 05m 29s	0.725000024
Running Trial #6	32	48	5	3	32	0.0067646	**0.769999981**	00h 05m 32s	**0.769999981**
Running Trial #7	64	16	3	2	64	0.00023916	0.714999974	00h 05m 34s	0.769999981
Running Trial #8	32	48	5	3	64	0.00026979	0.704999983	00h 05m 37s	0.769999981
Running Trial #9	64	48	3	2	96	0.00070791	0.595000029	00h 05m 39s	0.769999981
Running Trial #10	32	32	3	2	96	0.00022996	0.694999993	00h 05m 42s	0.769999981
Running Trial #11	64	48	5	3	96	0.0010584	0.720000029	00h 05m 45s	0.769999981
Running Trial #12	32	48	5	3	32	0.0067646	0.685000002	00h 05m 47s	0.769999981
Running Trial #13	32	64	5	3	2	0.001687	0.699999988	00h 05m 50s	0.769999981
Running Trial #14	**64**	**64**	**3**	**3**	**32**	**0.0098077**	**0.805000007**	**00h 05m 53s**	**0.805000007**
Running Trial #15	64	16	5	2	64	0.0046458	0.709999979	00h 05m 56s	0.805000007
Running Trial #16	32	32	5	3	32	0.0026357	0.725000024	00h 05m 59s	0.805000007
Running Trial #17	32	64	3	3	64	0.0013156	0.654999971	00h 06m 01s	0.805000007
Running Trial #18	64	64	3	2	96	0.0038373	0.694999993	00h 06m 04s	0.805000007
Running Trial #19	64	32	3	3	32	0.00067731	0.685000002	00h 06m 08s	0.805000007
Running Trial #20	64	64	3	3	32	0.0098077	0.790000021	00h 06m 12s	0.805000007
Running Trial #21	32	32	5	3	32	0.00026357	0.694999993	00h 06m 15s	0.805000007
Running Trial #22	64	48	3	2	64	0.00067106	0.74000001	00h 06m 20s	0.805000007
Running Trial #23	32	64	3	3	64	0.0045108	0.785000026	00h 06m 24s	0.805000007
Running Trial #24	32	64	3	3	64	0.00057886	0.745000005	00h 06m 28s	0.805000007
Running Trial #25	32	64	3	3	96	0.00064224	0.754999995	00h 06m 33s	0.805000007

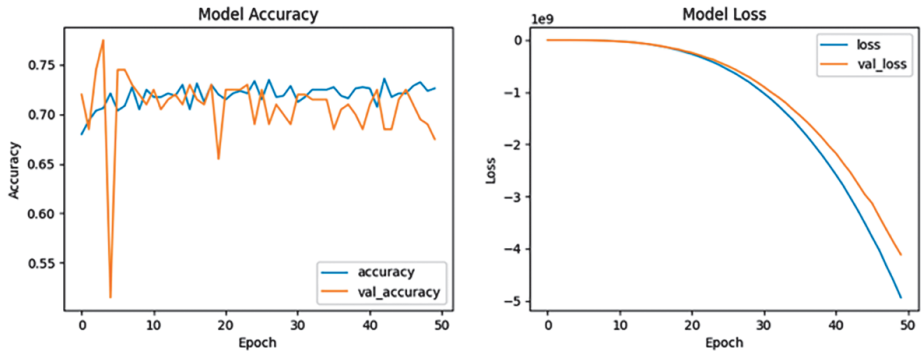

Figure 2. Experimental Results of the Proposed Model

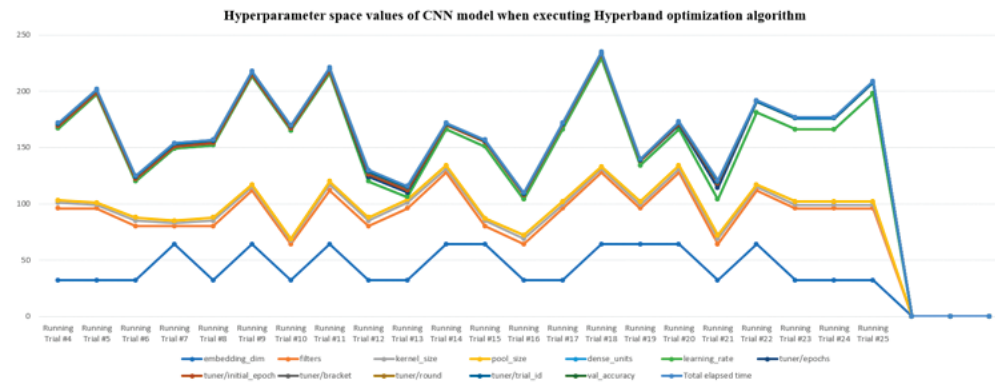

Figure 3. Hyperparameter Values of the CNN Model Obtained Using the Hyperband Optimization Algorithm

This comparison demonstrates that the proposed CNN model not only achieved superior performance but also benefited from automatic feature extraction without manual intervention. As a result, phishing detection capability was improved compared to traditional methods and recurrent neural network models.

The proposed CNN model is considered a robust solution for phishing detection due to its ability to identify complex data patterns from URL sequences. With an accuracy of 85.42%, the model not only matched but also outperformed results reported in previous studies. CNNs were shown to be particularly effective in automatically extracting features from raw data, thereby reducing the need for handcrafted feature engineering and making the model more robust to variations in input data. Compared to traditional approaches, significant performance improvements were achieved. This comparative analysis highlights the potential of CNNs in advancing phishing detection techniques and opens promising research directions in the field of cybersecurity.

Conclusion

The CNN model was shown to achieve strong performance in phishing detection due to its capability to automatically extract features from URLs. The hyperparameter optimization process using Keras Tuner was applied to improve key parameters such as embedding dimension, number of filters, and learning rate, thereby enhancing overall model effectiveness. However, one limitation was observed in the model's sensitivity to specific URL structures, which may affect accuracy when different phishing attack patterns are encountered. Future research may address this limitation by using more diverse datasets or by integrating CNNs with advanced architectures such as Transformers or RNNs to improve generalization and phishing detection capability. An accuracy of 80.50% was achieved, confirming the potential of CNNs in the automatic detection of malicious URLs and their contribution to enhanced cybersecurity.

REFERENCES

Chollet, F. 2021. "Keras Tuner: A hyperparameter tuning framework for Keras". Retrieved from https://keras.io/keras_tuner/

Gupta, S., Patel, R. H., & Chen, M. A. 2022. "Phishing detection using machine learning techniques: A comprehensive review". *Journal of Information Security, 11*(3), 134-152.

Huang, G., Liu, Z., & Maaten, L. V. D. 2020. "CNNs for phishing detection: A novel approach". *Journal of Machine Learning Research, 21*, 214-229.

Khan, K., & Hussain, A. M. 2021. "Detection of phishing websites using deep learning". *Journal of Cyber Security, 13*(2), 101-114.

Kim, Y. 2014. "Convolutional neural networks for sentence classification". *EMNLP*, 1746-1751.

Lee, J. H., Kim, H. J., & Lee, M. 2021. "Bidirectional LSTM networks for phishing detection". *Journal of Cyber Security*, 13(2), 115-128.

Li, L., Jamieson, K. G., DeSalvo, G., & Narayanaswamy, B. (2016). "Hyperband: A novel bandit-based approach to hyperparameter optimization". *Journal of Machine Learning Research, 18*, 1-52.

Mishra, A. R., & Sharma, S. 2020. "Dataseat for phishing URL detection". *Data Science Journal*, 17(4), 45-56.

Zhang, Y., & Yang, Q. 2020. "A survey on hyperparameter optimization methods and applications". *IEEE Transactions on Knowledge and Data Engineering*, 32(5), 1673-1685.

Zhang, Z., Zhao, J., & LeCun, Y. 2015. "Character-level convolutional networks for text classification". *Advances in Neural Information Processing Systems, 28*, 649-657.